Objectives:

- Practice using Recursion to traverse Linked Lists.
- Practice refactoring operations (refactoring iterative solutions to recursive implementations.)
- Working with projects with multiple files.
- Program comprehensions, debugging, and refactoring
- Practice using exceptions and exception handling in C++
- Practice using NodeList and Iterators.

Problem Description (Project 6) [30 Points]

In this project, you are given a partial implementation of a car rental system (a subsystem of a full system) that controls the vehicle's partial activities within the inventory. The system is expected to help the inventory automate its vehicle reservation and query processes (without getting involved in the customer information). It uses a data structure to keep track of the vehicle inventory in the system. Moreover, vehicle information is already provided in a text file, and no need to worry about inserting new vehicles into the file. However, the system needs to update the file after each reservation or return process. The good news is that most of the work is done on your behalf.

The given program performs at least the following operations:

1. Print All of the registered vehicles. (available and already reserved)
2. Print list of the available vehicles.
3. List of the reserved vehicles (rented currently by customers).
4. Check availability.
5. Reserve a vehicle.
6. return a reserved vehicle.
7. Print All of the registered vehicles to a text file.
8. Exit

For simplicity, only two major components (classes) were used to develop the system,: **Vehicle** and **Inventory** with a list of vehicles stored in a file.

An application file that contains the main function is provided as well. The main function controls the selection menu and reads the inventory information from the file (vehicles.txt).

Important Videos that explain how to work on (solve) this assignment:

1: https://web.microsoftstream.com/video/36a56f4b-1597-4111-9191-298cea363b98 (Links to an external site.)

2: https://web.microsoftstream.com/video/c20cdca4-bdc6-44c4-a6d8-7327e96833c5 (Links to an external site.)

3: https://web.microsoftstream.com/video/c5c7ae4c-0b3f-4ab4-918f-74107ece4aab (Links to an external site.)

Additionally, the following video (https://web.microsoftstream.com/video/98695059-c2d8-40c6-831c-a0a583772531 (Links to an external site.)) explains how the system is created and how it works to be able to understand it (comprehend the system) and then be able to do the refactoring operations explained below.

Data structures used in the system: dynamic Arrays

Your job:

Part 1: Refactoring: Replace the underlining dynamic arrays in classes Vehicle and Inventory with objects of NodeList class.

For the Vehicle class, a dynamic array (**string *otherFeaturesList;**) is used for storing the extra feature each vehicle has. This dynamic array needs to be replaced with an object of NodeList class for more flexibility as the dynamic array can not exceed the array capacity. With the NodeList object, as long as you have memory (RAM), you can add new vehicle objects. Of course, you need to update all of the affected methods by this replacement.

Same thing for the class Inventory. A dynamic array (**Vehicle* v_List;**) is used for storing the vehicles' information read from the file. This dynamic array needs to be replaced with an object of NodeList class for more flexibility as the dynamic array can not exceed the array capacity. With the NodeList object, as long as you have RAM, you can add new vehicle objects. Of course, you need to update all of the affected methods by this replacement (e.g., Inventory::push_Back method).

You can infer the missed methods from their signatures (prototypes) provided with the given code (see the header file). Please note that vehicles are compared based on their number of seats.

Part 2: Refactoring: reimplementing the following methods recursively (using **recursion**).

- void Inventory::printList()
- void Inventory::printResevedList()
- void Inventory::printAvailableList()
- bool Inventory::found(int seatsNo)

Part 3: In this part, you are expected to enhance (improve) the system with exceptions. That is, you need to handle the operations that can cause a problem during the runtime using exceptions. Below are some of the operations you need to improve by using exceptions:

Entering an invalid entry (other than 0 to 8)

File not found or can not be open for reading and/or writing).

Sending an invalid vehicle ID to the reservation method.

Trying to reserve an already reserved vehicle.

You need to show what other operations can be improved with exception handling.

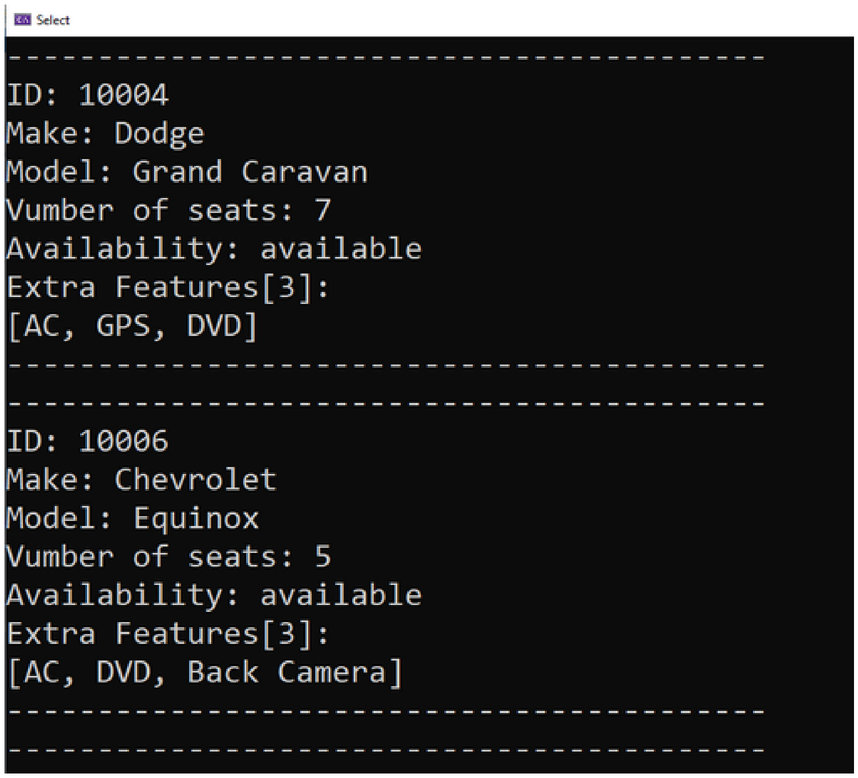
Supporting Code Files

- This project is similar to the prior project.
- A complete header file that you need to study before working on the implementation. The file contains two classes and relevant friend functions.
- The full implementation of a NodeList is posted on Canvas. It should be used to create underlining data structures in the second class (Inventory) for the vehicle list and in the first class (Vehicle) to hold the extra features.
- Full implementation for the main function with the operation menu and a simple test.
- A text file with sample records
- A zipped incomplete project is provided. Project6_OldVersion.zip Download Project6_OldVersion.zip

Clearing the console screen can be done using: system("cls");

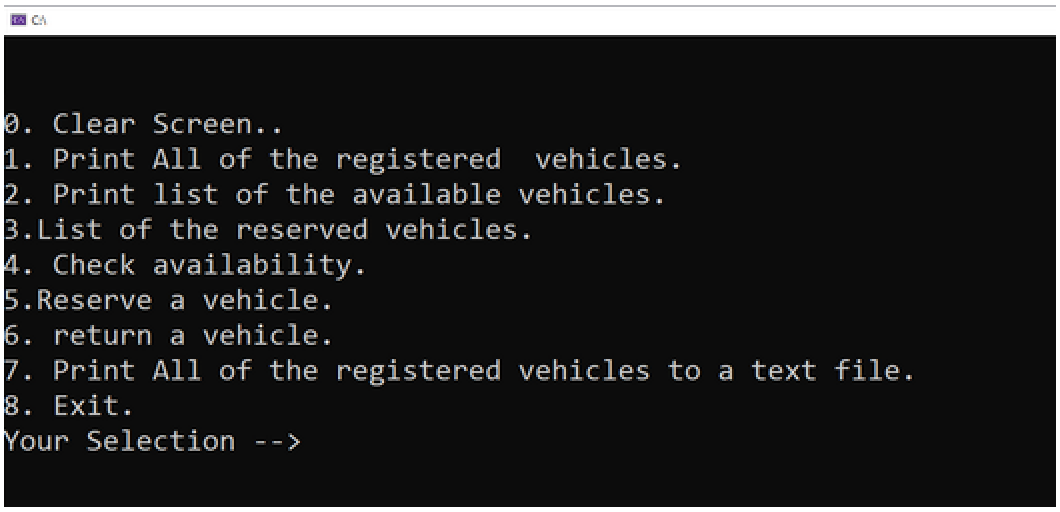
Example Screens:

This is how records are printed:



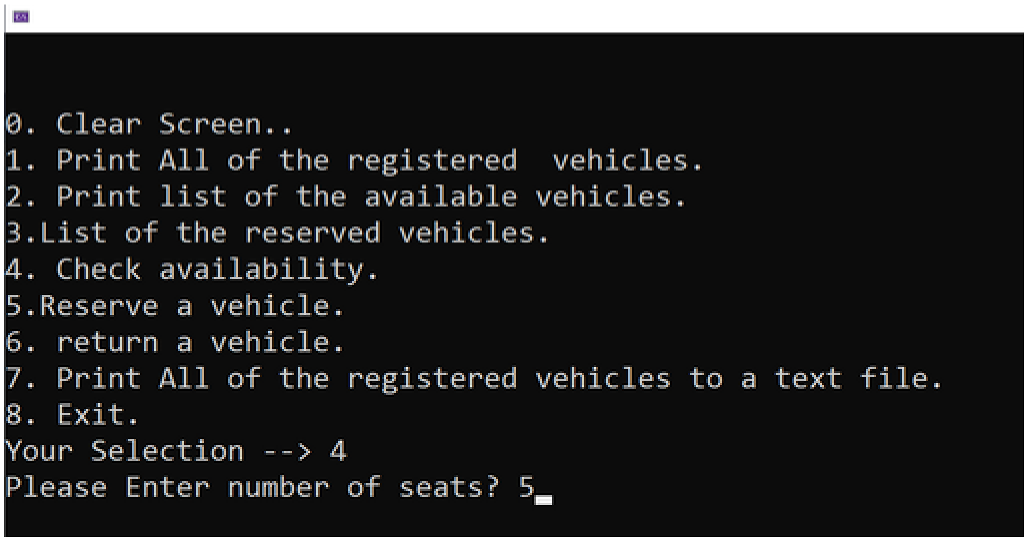
```
Select
-----------------------------------------
ID: 10004
Make: Dodge
Model: Grand Caravan
Vumber of seats: 7
Availability: available
Extra Features[3]:
[AC, GPS, DVD]
-----------------------------------------
-----------------------------------------
ID: 10006
Make: Chevrolet
Model: Equinox
Vumber of seats: 5
Availability: available
Extra Features[3]:
[AC, DVD, Back Camera]
-----------------------------------------
-----------------------------------------
```

Main Menue:



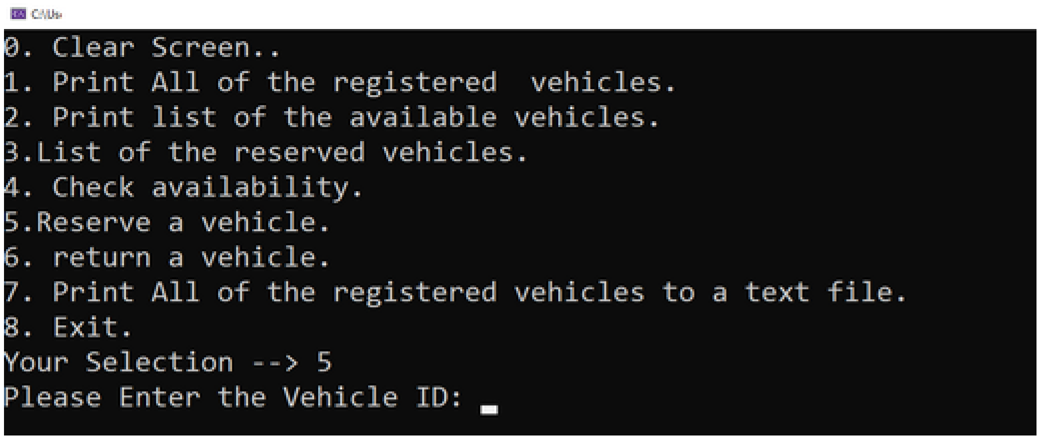
```
C:\

0. Clear Screen..
1. Print All of the registered  vehicles.
2. Print list of the available vehicles.
3.List of the reserved vehicles.
4. Check availability.
5.Reserve a vehicle.
6. return a vehicle.
7. Print All of the registered vehicles to a text file.
8. Exit.
Your Selection -->
```

Check Availability:



```
0. Clear Screen..
1. Print All of the registered  vehicles.
2. Print list of the available vehicles.
3.List of the reserved vehicles.
4. Check availability.
5.Reserve a vehicle.
6. return a vehicle.
7. Print All of the registered vehicles to a text file.
8. Exit.
Your Selection --> 4
Please Enter number of seats? 5_
```

Reserve A Vehicle:



```
0. Clear Screen..
1. Print All of the registered  vehicles.
2. Print list of the available vehicles.
3.List of the reserved vehicles.
4. Check availability.
5.Reserve a vehicle.
6. return a vehicle.
7. Print All of the registered vehicles to a text file.
8. Exit.
Your Selection --> 5
Please Enter the Vehicle ID: _
```

Useful Videos: Recordings:

https://web.microsoftstream.com/video/7e62a656-c46f-4bc9-8934-53be6bd014ba (Links to an external site.)

Similar video: (https://web.microsoftstream.com/video/98695059-c2d8-40c6-831c-a0a583772531 (Links to an external site.) **)**